# GetCache Monitor API User Guide

version 1.2.0

# Introduction

GetCache is an in-memory key-value store developed with .Net 4.5 very simple to use and manage.
GetCache is a distributed in-memory cache that supports data sharding on multiple nodes and data replication.
GetCache stores any type of data, objects can be serialized using JSON or XML, row object like images or videos can be stored as byte arrays.

GetCache Monitor API is a set of libraries that allow access to storage monitor interface.
External applications can use these API to check the nodes states and to perform commands.

# .Net Client library

## Distribution

GetCache Monitor API library can be downloaded from the website www.getcache.net or via NuGet.
The package contains the DLL file *GetCache.Client.Monitor.dll* .

The .Net Monitor API is also distributed with the server package.

The .Net Monitor API is developed using .Net 4.5 as the GetCache Server.

## Monitor API

The class that exposes the client API is **GetCacheMonitorClient** that implements the interface **IMonitorClient**. The API offers methods to operate with the remote GetCache server.
GetCacheMonitorClient can connect a single node or a cluster of nodes.

## API List

***List<NodeDescriptor> GetClusterNodeList()***
List the available nodes of the cluster.

***ServerState GetServerState(Guid nd);***
Get the server state and the installed modules list of the node that has the GUID passed in input.

***MonitorResult DeployModule(Guid nd, ModulePackage package);***
Deploy an application module (Robot) on the server. The method operates on the server node

that has the GUID passed in input. The ModulePackage input class is structured as the module files and directories. This class is a transport for DLL filename and body and the module directories. The GetCacheMonitorClient class offers the utility method *CreateModulePackage* that can be used to produce the ModulePackage given the path of the directory that contains all the module's files.

***MonitorResult UndeployModule(Guid nd, String moduleName, bool removeFile)***
Undeploy a module installed on the server. The method operates on the server node that has the GUID passed in input. If the removeFile input parameter is true the files of the module are deleted from the server.

***MonitorResult ProcessConsoleCommand(Guid nd, ConsoleCommand cmd)***
Process a remote console command on the node. The commands are the available command-line console commands and can have the same arguments.

***Boolean IsNodeAlive(Guid nd)***
Check if the node is running.

### Connect the server

To connect the cluster, create the GetCacheMonitorClient object giving it a staring list of nodes (at least one node must be provided)

Example of client initialization:

```
// Create the monitor configuration and connect the cluster
MonitorConfig config = new MonitorConfig();
config.AddNode("localhost", 1010);
GetCacheMonitorClient client = new GetCacheMonitorClient(config);
```

After the client has been created, you can retrieve the list of nodes in the cluster

```
List<NodeDescriptor> nodes = client.GetClusterNodeList();
```

then you can proceed by querying the status of each node in the cluster

```
foreach (NodeDescriptor nd in nodes)
{
    if (client.IsNodeAlive(nd.NodeGuid))
    {
        ServerState state = client.GetServerState(nd.NodeGuid);
```

```
        Console.WriteLine(String.Format("Node: {0} - {1}", nd.Name, nd.NodeGuid));
        Console.WriteLine(String.Format("Keys Count: {0}", state.KeysCount));
        Console.WriteLine(String.Format("Module count: {0}",
                state.Modules!=null?state.Modules.Count:0));
    }
}
```

## Examples of usage

### Deploy a module on a remote node

```
MonitorConfig config = new MonitorConfig();
config.AddNode("localhost", 1010);
GetCacheMonitorClient client = new GetCacheMonitorClient(config);

List<NodeDescriptor> nodes = client.GetClusterNodeList();

string path = ".\\testdeploy\\MyList";
ModulePackage package = client.CreateModulePackage(path);

if (package != null)
{
   MonitorResult result = client.DeployModule(nodes[0].NodeGuid, package);
   Console.WriteLine(String.Format("{0} - {1}", result.ResultCode, result.ResultDesc));
}
```

### Undeploy a module on a remote node

```
MonitorConfig config = new MonitorConfig();
config.AddNode("localhost", 1010);
GetCacheMonitorClient client = new GetCacheMonitorClient(config);

List<NodeDescriptor> nodes = client.GetClusterNodeList();

MonitorResult result = client.UndeployModule(nodes[0].NodeGuid, "MyList", true);
Console.WriteLine(String.Format("{0} - {1}", result.ResultCode, result.ResultDesc));
```

### Execute a remote console command

```
MonitorConfig config = new MonitorConfig();
config.AddNode("localhost", 1010);
GetCacheMonitorClient client = new GetCacheMonitorClient(config);

List<NodeDescriptor> nodes = client.GetClusterNodeList();

ConsoleCommand cmd = new ConsoleCommand() {
Command = "show",
Arguments = new List<string>(),
};
cmd.Arguments.Add("robots");

MonitorResult result = client.ProcessConsoleCommand(nodes[0].NodeGuid, cmd);
Console.WriteLine(String.Format("{0} - {1}", result.ResultCode, result.ResultDesc));
Console.WriteLine(String.Format("{0}", result.ResultOutput));
```